Bücher mit Code online publizieren

Gert-Ludwig Ingold

https://github.com/gertingold/lit2024

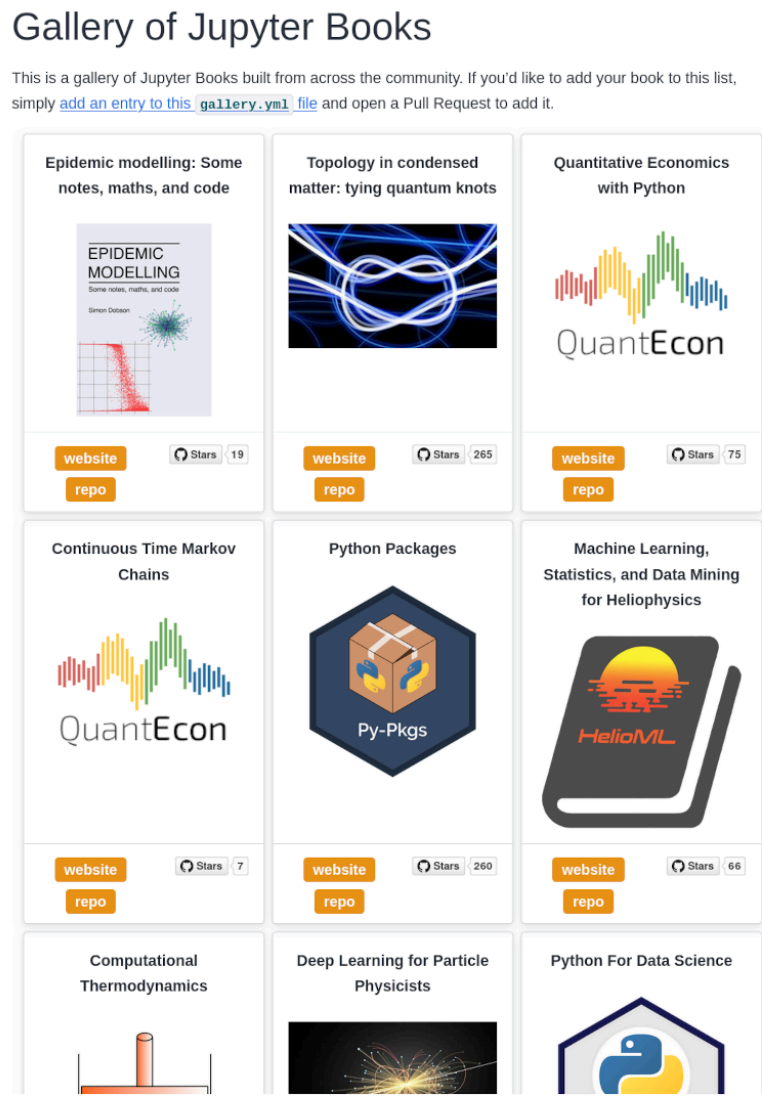
Anwendungsszenarien

Mein Anwendungsszenario

- Manuskript zu einem Programmierkurs frei zugänglich machen
- Präsentation auf einer responsiven Webseite
- Code wird zum Aufbau der Inhalte der Webseite ausgeführt und die Resultate werden integriert

Allgemeinere Szenarien

- allgemeine Lehrmaterialen mit Code zum Erzeugen von Abbildungen oder ganz ohne Code
- wissenschaftliche Arbeiten, z.B. mit Datenanalyse



Lecture notes »Tools for Scientific Computing«

Sphinx

www.sphinx-doc.org

- in erster Linie zur Erzeugung von Dokumentation
- auch schöne PDF-Ausgabe
- Markup mit reStructuredText
- zahlreiche Erweiterungsmöglichkeiten
- Sprache einstellbar

- standardmäßig keine Codeausführung mit Einbettung des Ergebnisses
- Webseiten nicht responsiv

Beispiel links: gertingold.github.io/tools4scicomp

Aspekte werden wir später auf die von NumPy zur Verfügung gestellten Arrays übertragen können.

Listen sind uns beispielsweise bereits in Kapitel 4.1 begegnet, wo wir die `range()`-Funktion verwendet hatten, um einen Schleifenzähler mit Werten zu versorgen. Dabei werden die benötigten Werte nur bei Bedarf erzeugt. Um alle Werte auf einmal zu sehen, hatten wir die `list()`-Funktion verwendet und dabei eine Liste erzeugt.

```
meine_liste = list(range(20))
print(meine_liste)
print(type(meine_liste))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
<class 'list'>
```

Mit der zweiten Ausgabezeile wird hier nachgewiesen, dass der Datentyp des Objekts `meine_liste` tatsächlich eine Liste ist.

Wenn man die Länge einer Liste nicht kennt, kann man diese mit Hilfe der `len()`-Funktion bestimmen.

```
liste1 = list(range(1, 17, 3))
print(f'Länge der ersten Liste:  {len(liste1)} Elemente')
liste2 = ['Stein', 'Papier', 'Schere']
print(f'Länge der zweiten Liste: {len(liste2)} Elemente')
```

```
Länge der ersten Liste:  6 Elemente
Länge der zweiten Liste: 3 Elemente
```

Jupyter Book

jupyterbook.org | executablebooks.org

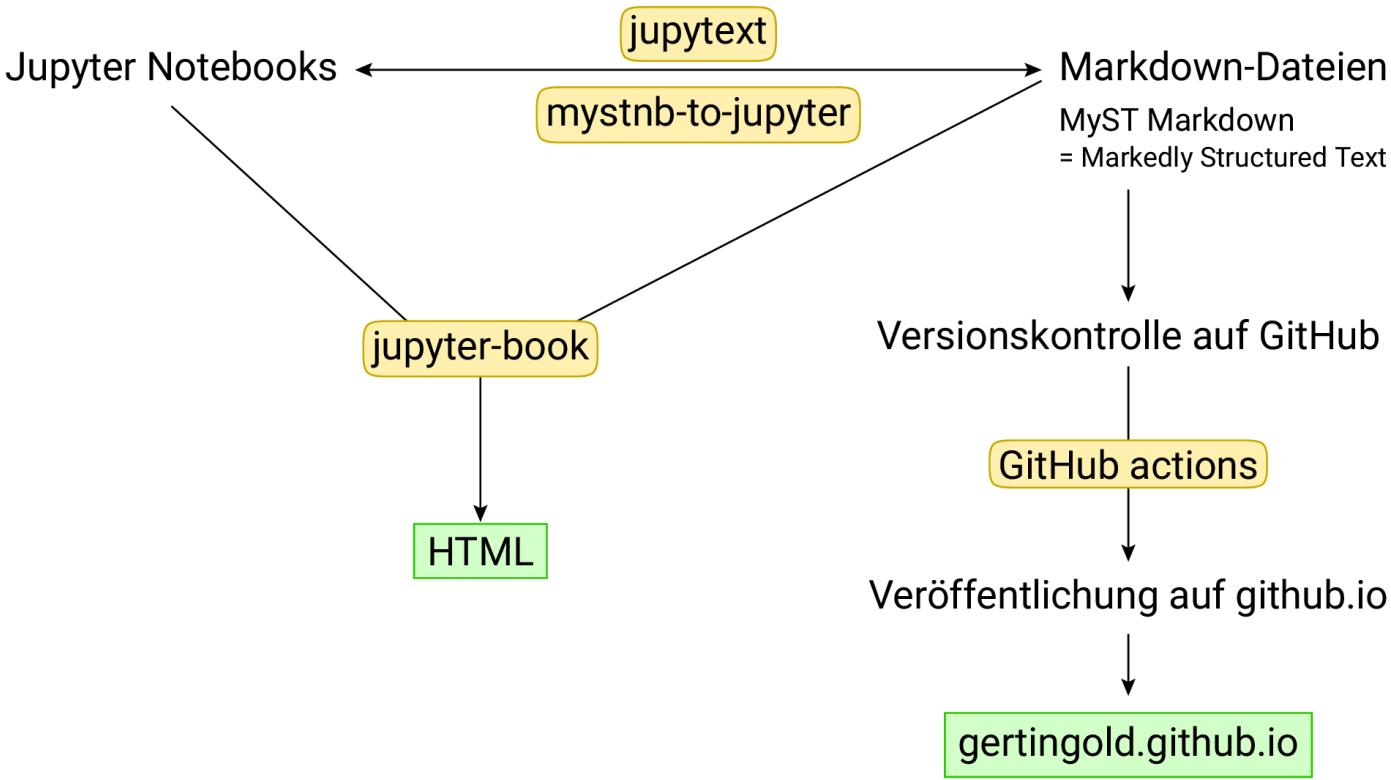
- basiert unter anderem auf Sphinx
- Sprache über die Sphinx-Konfiguration festlegbar

  ```
  Listen sind uns beispielsweise bereits in
  {numref}`forloop` begegnet, wo wir die
  ```

- Ergebnis der Ausführung von Code lässt sich einbetten
- responsive Webseite

Beispiel links: gertingold.github.io/epriprog

Ein Gesamtüberblick



Harmonischer Oszillator

Zeitunabhängige Schrödingergleichung

$$-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}x^2}\psi(x) + \frac{m}{2}\omega^2 x^2 \psi(x) = E\psi(x)$$

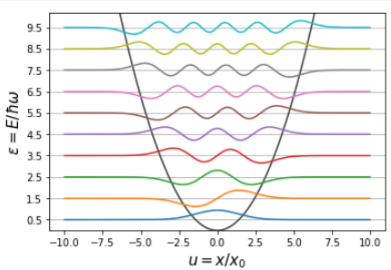
[1]:
```
from math import factorial, pi, sqrt

import numpy as np
import numpy.linalg as LA
from scipy.special import eval_hermite
import matplotlib.pyplot as plt

import ipywidgets as widgets
from ipywidgets import interact
```

[2]:
```
def ho_numerical(n, delta):
    u = delta*np.arange(-n, n+1)
    hamilton = -(np.eye(2*n+1, k=1) - 2*np.eye(2*n+1) + np.eye(2*n+1, k=-1))/delta**2
    hamilton = hamilton + np.diag(u**2/4)
    eigvals, eigvecs = LA.eigh(hamilton)
    return u, eigvals, eigvecs

n = 1000
delta = 0.01
u, eigvals, eigvecs = ho_numerical(n, delta)
plt.plot(u, u**2/4, '#3f3f3f')
plt.ylim(0, 10.2)
for n in range(10):
    psi = 7*eigvecs[:, n]/LA.norm(eigvecs[:, n])
    plt.plot(u, eigvals[n]+psi)
plt.xlabel('$u = x/x_0$', fontsize=15)
plt.ylabel(r'$\epsilon = E/\hbar\omega$', fontsize=15)
plt.yticks(np.arange(0.5, 10, 1))
plt.grid(axis='y')
```



Jupyter-Notebooks

jupyter.org

- Integration von Code, auch mit interaktiver Kontrolle, sowie Text, Formeln und Multimediaelementen
- Anbindung an Python-, R-, Julia-Kernel und viele mehr
- Einsatz in Lehre, Datenanalyse und vielem mehr

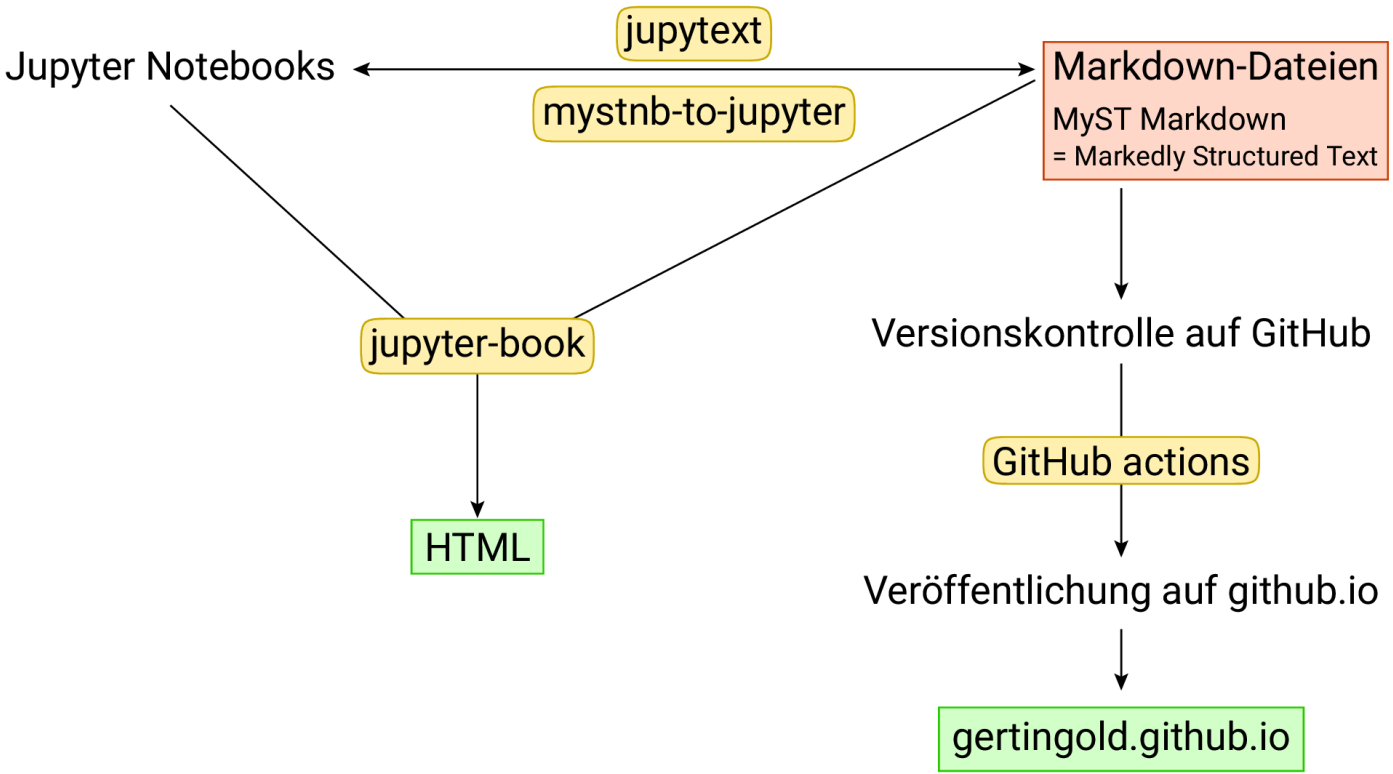
- Möchte man in der Lehre mit vorgefertigtem Code arbeiten oder eher Code entwickeln?
- Markdown, aber MyST Markdown nur mit einer Extension

Speicherung des Jupyter-Notebooks im JSON-Format

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Harmonischer Oszillator"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Zeitunabhängige Schrödingergleichung\n",
    "\n",
    "$$-\\frac{\\hbar^2}{2m}\\frac{\\text{d}^2}{\\text{d}x^2}\\psi(x)+\\frac{m}{2}\\omega^2x^2\\psi(x) = E\\psi(x)$$"
```

- Versionskontrolle kann Schwierigkeiten bereiten
- Verwendung von `nbstripout`-Filter

MyST Markdown-Dateien als Basis



CommonMark

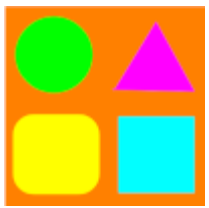
commonmark.org

Variante 1	Variante 2	Ausgabe
`# Überschrift 1`	`Überschrift 1` `==========`	# Überschrift 1
`## Überschrift 2`	`Überschrift 2` `----------`	## Überschrift 2
`##### Überschrift 5`		Überschrift 5
`*kursiv*`	`_kursiv_`	*kursiv*
`**fett**`	`__fett__`	**fett**
`` `inline` ``		`inline`

CommonMark

Variante 1	Variante 2	Ausgabe
`* Eintrag 1` `* Eintrag 2`	`- Eintrag 1` `- Eintrag 2`	▪ Eintrag 1 ▪ Eintrag 2
`1. Eintrag 1` `2. Eintrag 2`	`1) Eintrag 1` `2) Eintrag 2`	1. Eintrag 1 2. Eintrag 2
`---`	`***`	---
`> Blockzitat`		> Blockzitat

CommonMark

Variante 1	Variante 2	Ausgabe
`[Link](http://a.org)`	`[Link][1]` ⋮ `[1]: http://a.org`	Link
`![Bild](images/bild.png)`	`![Bild][1]` ⋮ `[1]: images/bild.png`	
`` ``` ```` `a = 3` `print(a**2)` ```` ``` ````	`␣␣␣␣a = 3` `␣␣␣␣print(a**2)`	`a = 3` `print(a**2)`

MyST Markdown anhand von Beispielen

MyST Markdown erweitert CommonMark jupyterbook.org/en/stable/content/

```
es kann auch Fälle geben, in denen man selbst programmieren muss, zum Beispiel wenn man eine Exponentialfunktion
auf einen Waschbären anwenden möchte ({numref}`fig:exponential_raccoon`).
```{figure} images/einleitung/exponential_raccoon.png
---
width: 10cm
name: fig:exponential_raccoon
---
Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).
```
```

Bildbearbeitungsaufgaben sind geeignete Programme verfügbar, aber es kann auch Fälle geben, in denen man selbst programmieren muss, zum Beispiel wenn man eine Exponentialfunktion auf einen Waschbären anwenden möchte (Abb. 1.1).

Abb. 1.1 Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).

MyST Markdown anhand von Beispielen

````
````{margin}
```{admonition} Tipp
:class: tip
Eine größere Darstellung von Abbildung können Sie durch Klicken auf das entsprechende Bild erhalten.
```
````

Das Beispiel der Digitalkamera schlägt die Brücke vom Alltag ins Labor. Im Grunde genommen ist eine Digitalkamera nichts anderes als ein Messgerät, das
````

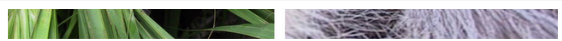


Abb. 1.1 Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).

Das Beispiel der Digitalkamera schlägt die Brücke vom Alltag ins Labor. Im Grunde genommen ist eine Digitalkamera nichts anderes als ein Messgerät, das optische Information in digitale Daten umwandelt und zur Verfügung stellt, sei es als Rohdaten oder in einer vorverarbeiteten Form. Im Labor wird es erforderlich sein, mit dem Messgerät zu kommunizieren, um Messdaten herunterzuladen. Häufig geschieht dies mit einem vom Hersteller zur Verfügung gestellten Programm, aber man kann sich auch vorstellen, dass man die Schnittstelle des Messgeräts, über die die Messdaten zur Verfügung gestellt werden, mit eigenem Programm anspricht.

Tipp

Eine größere Darstellung von Abbildung können Sie durch Klicken auf das entsprechende Bild erhalten.

MyST Markdown anhand von Beispielen

````
```{admonition} Frage
Was ergibt `-2*4+3**2`? Was ergibt `6**4//2`?
```
````

Frage

Was ergibt `-2*4+3**2`? Was ergibt `6**4//2`?

````
```{admonition} Hinweis
:class: tip
Seit Python 3.9 lässt sich die kleinste darstellbare
Zahl, also `5e-324` mit Hilfe von `math.ulp(0)`
erhalten.
```
````

Hinweis

Seit Python 3.9 lässt sich die kleinste darstellbare Zahl, also `5e-324` mit Hilfe von `math.ulp(0)` erhalten.

````
````{admonition} Weiterführendes (rechts aufklappen)
:class: toggle
Nach der Kompilierung des obigen C-Programms entsteht al
ein so genanntes Assembler-Programm, das schon sehr masc
von einem Assembler in den von einem Computer les- und a
Maschinencode umgewandelt wird.
```{code-block}
            .file   "bsp_exp.c"
            .text
⋮
````

Weiterführendes (rechts aufklappen) Click to show

MyST Markdown anhand von Beispielen

```
- Python besitzt eine umfangreiche Standardbibliothek (»Python comes with batteries included«) und das
  wissenschaftliche Rechnen wird durch eine Vielzahl freier Programmbibliotheken, wie {program}`NumPy/SciPy`, das wir
  im Kapitel {ref}`scipy` besprechen werden, unterstützt.
- Python hat sich in den letzten Jahren zu einer sehr populären Sprache entwickelt, unter anderem im Bereich der
  wissenschaftlichen Datenanalyse.

Bei den beiden physikalischen Beobachtungen, die in den letzten Jahren die Aufmerksamkeit einer breiten Öffentlichkeit
erregten, nämlich die Beobachtung von Gravitationswellen [^prd93] und die Aufnahme des Abbilds eines schwarzen
⋮
[^prd93]: B. P. Abbott  et al. , [Phys. Rev. D  93 , 122003 (2016)](https://doi.org/10.1103/PhysRevD.93.122003). Das
Analysepaket PyCBC basiert auf Python und auch das Analysepaket GstLAL enthält einige Pythonskripte.
```

- Python besitzt eine umfangreiche Standardbibliothek (»Python comes with batteries included«) und das wissenschaftliche Rechnen wird durch eine Vielzahl freier Programmbibliotheken, wie **NumPy/SciPy**, das wir im Kapitel Numerische Programmbibliotheken am Beispiel von NumPy/SciPy besprechen werden, unterstützt.
- Python hat sich in den letzten Jahren zu einer sehr populären Sprache entwickelt, unter anderem im Bereich der wissenschaftlichen Datenanalyse.

Bei den beiden physikalischen Beobachtungen, die in den letzten Jahren die Aufmerksamkeit einer breiten Öffentlichkeit erregten, nämlich die Beobachtung von Gravitationswellen [[3]] und die Aufnahme des Abbilds eines schwarzen Loches (Abb. 1.2) [[4]], spielte Python bei der

MyST Markdown anhand von Beispielen

```
Im ersten Beispiel soll der Zusammenhang zwischen der Winkelgeschwindigkeit $\omega$
und der Beschleunigung $a$ bei einer Rotationsbewegung untersucht werden. Gemäß
der Mechanik erfährt ein Objekt im Abstand $r$ von der Drehachse die Beschleunigung

$$a = r\omega^2\,.$$

Zur experimentellen Untersuchung wird ein Smartphone wie in
```

Im ersten Beispiel soll der Zusammenhang zwischen der Winkelgeschwindigkeit ω und der Beschleunigung a bei einer Rotationsbewegung untersucht werden. Gemäß der Mechanik erfährt ein Objekt im Abstand r von der Drehachse die Beschleunigung

$$a = r\omega^2\,.$$

Zur experimentellen Untersuchung wird ein Smartphone wie in Abb. 2.1 in einer Salatschleuder montiert, und diese in eine Drehung versetzt. Die im Smartphone vorhandenen Sensoren

Warnung

Beim Experimentieren mit dem Smartphone ist Vorsicht geboten, um mechanische Beschädigungen auszuschließen. Das

MyST Markdown anhand von Beispielen

````
```{code-block} python
---
linenos: true
emphasize-lines: 18,21-25,29
---
import random

def get_result(n_self, n_other):
    result = (n_self - n_other) % 3
    if result == 0:
        return 'Das Spiel endete unentschieden.'
elif result == 1:
    return 'Du hast gewonnen.'
return 'Du hast leider verloren.'

objekte = ['Stein', 'Papier', 'Schere']
info_text = ('\n[0] Stein\n'
             '[1] Papier\n'
             '[2] Schere\n\n'
             'Gib eine Zahl zwischen 0 und 2 ein oder -1

while True:
    n_benutzer = int(input(info_text))
     ⋮
```
````

```
 1 import random
 2
 3 def get_result(n_self, n_other):
 4     result = (n_self - n_other) % 3
 5     if result == 0:
 6         return 'Das Spiel endete unentschieden.'
 7     elif result == 1:
 8         return 'Du hast gewonnen.'
 9     return 'Du hast leider verloren.'
10
11 objekte = ['Stein', 'Papier', 'Schere']
12 info_text = ('\n[0] Stein\n'
13              '[1] Papier\n'
14              '[2] Schere\n\n'
15              'Gib eine Zahl zwischen 0 und 2 ein oder -1 zum Beenden: ')
16
17 while True:
18     n_benutzer = int(input(info_text))
19     if 0 <= n_benutzer <= 2:
20         n_computer = random.randrange(3)
21         print('-'*40)
22         print(f'Benutzer: {objekte[n_benutzer]}')
23         print(f'Computer: {objekte[n_computer]}')
24         print(get_result(n_benutzer, n_computer))
25         print('-'*40)
26     elif n_benutzer == -1:
27         break
28     else:
29         print('\n*** ungültige Eingabe\n')
```

MyST Markdown anhand von Beispielen

````
Möchte man aus den Werten zweier Variablen eine komplexe Zahl konstruieren, geht dies mit der zweiten der gerade genannten Methoden sehr einfach
```{code-cell} python
x = 1
y = 2
z1 = complex(x, y)
z2 = complex(x, -y)
z1/z2
```

Falls man die Funktion {func}`complex` nicht verwenden möchte, muss man beachten, dass die folgenden beiden Wege
````

Möchte man aus den Werten zweier Variablen eine komplexe Zahl konstruieren, geht dies mit der zweiten der gerade genannten Methoden sehr einfach

```
x = 1
y = 2
z1 = complex(x, y)
z2 = complex(x, -y)
z1/z2
```

```
(-0.6+0.8j)
```

Falls man die Funktion `complex()` nicht verwenden möchte, muss man beachten, dass die

MyST Markdown anhand von Beispielen

````
```{code-cell} python
---
tags: [raises-exception]
---
x = 18
y = 9
z = x+yj
```
````

```
x = 18
y = 9
z = x+yj
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[62], line 3
      1 x = 18
      2 y = 9
----> 3 z = x+yj

NameError: name 'yj' is not defined
```

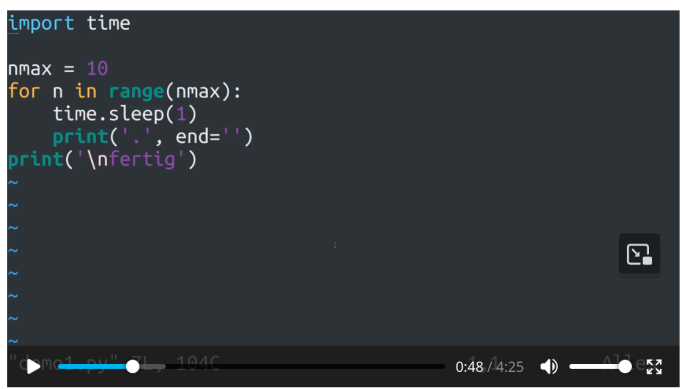
MyST Markdown anhand von Beispielen

```
auch im nächsten Abschnitt noch einmal eine Rolle spielen, wenn wir die Ausgabe von Daten in eine Datei besprechen.

<video width="640" height="360" controls>
  <source src="https://gertingold.github.io/resources/flush.webm" type="video/webm">
Ihr Browser unterstützt nicht das video-Tag.
</video>

(readfile)=
## Lesen von Dateien
```

auch im nächsten Abschnitt noch einmal eine Rolle spielen, wenn wir die Ausgabe von Dater
in eine Datei besprechen.



7.2. Lesen von Dateien

_conf.yml

```yaml
title          : Einführung in Prinzipien der Programmierung
author         : Gert-Ludwig Ingold
copyright      : "CC-BY"
logo           : ""

only_build_toc_files: true

repository:
  url: https//github.com/gertingold/epriprog
  path_to_book: epriprog

html:
  use_repository_button: false
  use_issues_button: false
  use_edit_page_button: false
  home_page_in_navbar: false

latex:
  latex_documents:
    targetname: epriprog.tex
  latex_elements:
    papersize: a4paper
```

_toc.yml

```yaml
root: intro
options:
  numbered: true
chapters:
  - file: einleitung
  - file: vorschau
  - file: datentypen
  - file: kontrollstrukturen
  - file: funktionen
  - file: sequenzen
  - file: einausgabe
  - file: scipy
  - file: objektorientiert
  - file: zahlensysteme
  - file: floats
  - file: unicode
```

3. Einfache Datentypen, Variablen und Zuweisungen

In Kapitel 2 hatten wir bereits verschiedene Datentypen kennengelernt. Hierzu gehören ganze Zahlen, die zum einen durch die explizite Umwandlung der Benutzereingabe in eine ganze Zahl oder als Ergebnis der `randrange()`-Funktion erzeugt wurden. Des Weiteren kamen Wahrheitswerte vor. Ganz explizit war dies bei `True` der Fall, das zur Konstruktion einer Dauerschleife verwendet wurde.

Diese beiden Datentypen, ganze Zahlen und Wahrheitswerte gehören zu den einfachen Datentypen. Daneben gibt es auch zusammengesetzte Datentypen, für die wir in Kapitel 2 ebenfalls schon Beispiele gesehen hatten. Dazu gehören die Zeichenketten, zum Beispiel in den Zeilen 12 bis 15 des Beispielprogramms, aber auch die durch eckige Klammern gekennzeichnete Liste in Zeile 11.

In diesem Kapitel werden wir uns zunächst den einfachen Datentypen zuwenden und die zusammengesetzten Datentypen in einem späteren Kapitel besprechen. Dabei werden wir vor allem neben den ganzen Zahlen noch weitere numerische Datentypen kennenlernen, die für natur- und ingenieurwissenschaftliche Anwendungen große Bedeutung besitzen.

3.1. Ganze Zahlen

Wir beginnen bei der Besprechung der numerischen Datentypen mit den ganzen

```
program datentyp
  integer :: n
  real :: x

  n = 2
  x = n
  write(*, *) n, x
end program datentyp
```

erzeugt die Ausgabe

```
           2   2.00000000
```

Gemäß der Deklaration zu Beginn des Programms handelt es sich bei der Variable `n` um einen Integer, während `x` ein Float ist, der in Fortran mit `real` bezeichnet wird. In der hervorgehobenen Zeile findet bei der Zuweisung automatisch

einfach abgeschnitten wird:

```
int(2.7)
```

```
2
```

Bereits das Anhängen eines Punktes genügt, damit Python die Zahl als Float interpretiert:

```
print(type(2.))
```

```
<class 'float'>
```

Für Floats gibt es zwei mögliche Schreibweisen. Zum einen die kann man die Dezimalbruchschreibweise verwenden,

GitHub

github.org

- Plattform für Software-Versionskontrolle und Projektzusammenarbeit
- freier persönlicher Account

github.io

- Webseiten unter *username*.github.io unter Verwendung eines öffentlichen GitHub-Repositories mit diesem Namen
- Erzeugung des Inhalts aus anderen GitHub-Repositories mit Hilfe von GitHub-Actions in einem Zweig `gh-pages`

- statische Webseiten, also z.B. keine PHP-Skripten
- Beschränkung der Seitengröße und Bandbreite, nicht für kommerziellen Gebrauch

GitHub Actions

`.github/workflows/deploy.yml`

```yaml
      uses: actions/setup-python@v5
      with:
        python-version: '3.11'

    - name: install dependencies
      run: |
        conda install -y python=3.11
        conda env update --file environment.yml --name base
        echo $CONDA/bin >> $GITHUB_PATH

    - name: build the book
      run: |
        jupyter-book build epriprog

    - name: deploy book's HTML to gh-pages branch
      uses: peaceiris/actions-gh-pages@v4
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: epriprog/_build/html
```

`environment.yml`

```yaml
name: epriprog
channels:
  - default
  - conda-forge
dependencies:
  - python==3.11
  - scipy
  - matplotlib
  - jupyter-book
```

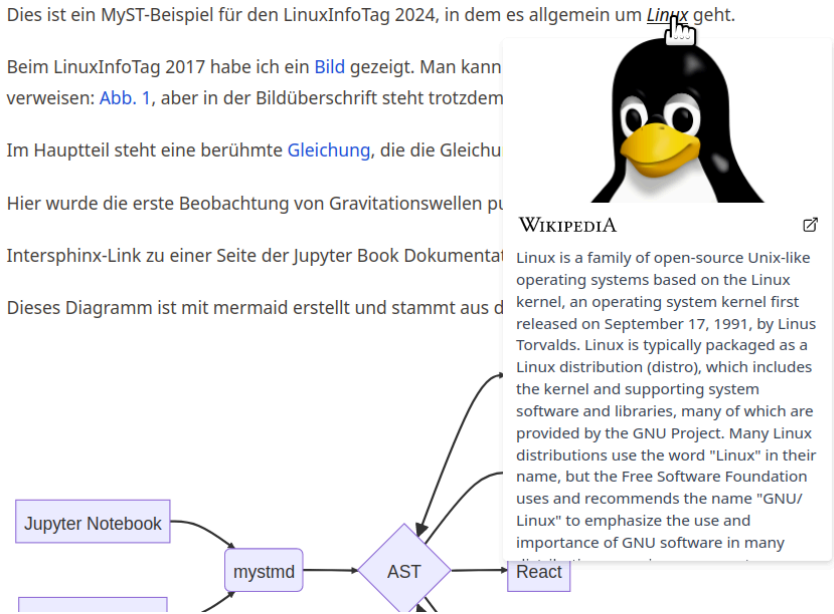
mystmd

mystmd.org

- neben Jupyter Book zweites (und jüngeres) Hauptprojekt von executable{books}
- Javascript statt Python, schneller und mehr Funktionalität
- Überlapp mit Jupyter Book
  - Jupyter Book Quellen verwendbar
  - Intersphinx für Verlinkung mit Sphinx-Dokumentationen
- vor allem Templates für Journale (derzeit 2 `docx`, 18 `tex`, 2 `typst`) sowie 2 Templates für Webseiten (article und book)
- Möglichkeit für interaktive Inhalte

- Lokalisierung??

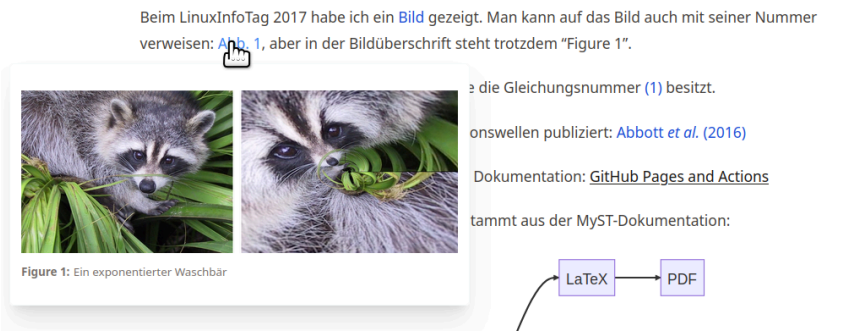
Ein paar Eindrücke von `mystmd`



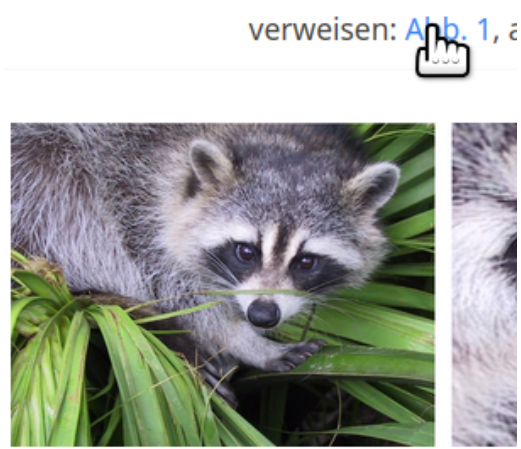

```
Dies ist ein MyST-Beispiel für den LinuxInfoTag 2024, in
dem es allgemein um <wiki:Linux> geht.
```

Beispielcode, siehe github.com/gertingold/lit2024/tree/main/myst-example

Ein paar Eindrücke von `mystmd`




```
Beim LinuxInfoTag 2017 habe ich ein
[Bild](#exponential_raccoon) gezeigt. Man kann auf das
Bild auch mit seiner Nummer verweisen:
[Abb. %s](#exponential_raccoon), aber in der
Bildüberschrift steht trotzdem "Figure 1".
```



Ein paar Eindrücke von `mystmd`

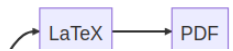
Im Hauptteil steht eine berühmte Gleichung, die die Gleichungsnummer (1) besitzt.

Hier w … ott *et al.* (2016)

$$E = mc^2 \tag{1}$$

Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation: GitHub Pages and Actions

Dieses Diagramm ist mit mermaid erstellt und stammt aus der MyST-Dokumentation:



`01-einleitung.md`

```
Im Hauptteil steht eine berühmte [Gleichung](#einstein),
die die Gleichungsnummer {numref}`einstein` besitzt.
```

Hauptteil

Figure 1: Ein exponentierter Waschbär

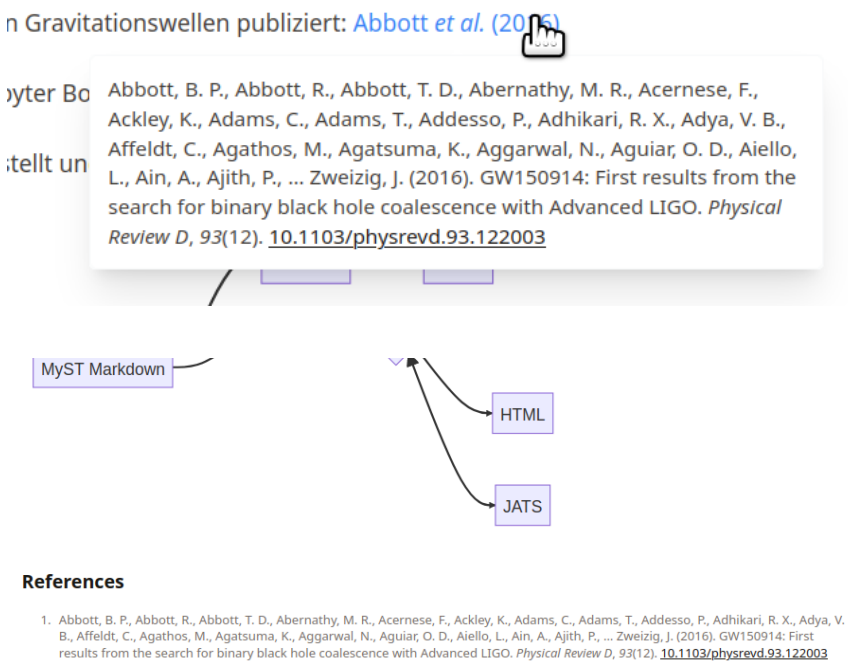
Dies ist eine Gleichung:

$$E = mc^2 \tag{1}$$

`02-hauptteil.md`

````
Dies ist eine Gleichung:
```{math}
:label: einstein
E = mc^2
```
````

Ein paar Eindrücke von `mystmd`



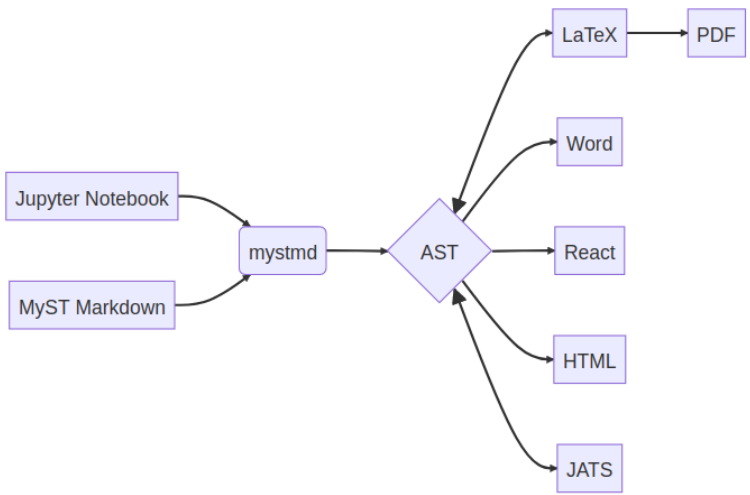

```
Hier wurde die erste Beobachtung von Gravitationswellen
publiziert: @doi:10.1103/PhysRevD.93.122003
```

- Am Ende wird automatisch eine Referenzenliste erzeugt.

Ein paar Eindrücke von `mystmd`

Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation: GitHub Pages and Actions

Dieses Diagramm ist mit mermaid erstellt und stammt aus der MyST-Dokumentation:




```
Intersphinx-Link zu einer Seite der Jupyter Book
Dokumentation: [](myst:jupyterbook#publish/gh-pages)

Dieses Diagramm ist mit mermaid erstellt und stammt aus
der MyST-Dokumentation:

```{mermaid}
flowchart LR
  A[Jupyter Notebook] ⟶ C
  B[MyST Markdown] ⟶ C
  C(mystmd) ⟶ D{AST}
  D ⟷ E[LaTeX]
  E ⟶ F[PDF]
  D ⟶ G[Word]
  D ⟶ H[React]
  D ⟶ I[HTML]
  D ⟷ J[JATS]
```
```

Nützliche Links

- Executable Book Project: executablebooks.org
- Jupyter Book: jupyterbook.org
- MyST: mystmd.org

- Quellen zu diesem Vortrag: github.com/gertingold/lit2024

- Beispiel mit Jupyter Book: github.com/gertingold/epriprog
- Beispiel mit Sphinx: github.com/gertingold/tools4scicomp

Fragen?